EXPERIMENT - 29

Object Oriented Programming Lab

Aim

Write a program to define the function template for swapping two items of various datatypes such as integers, float and characters.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 29

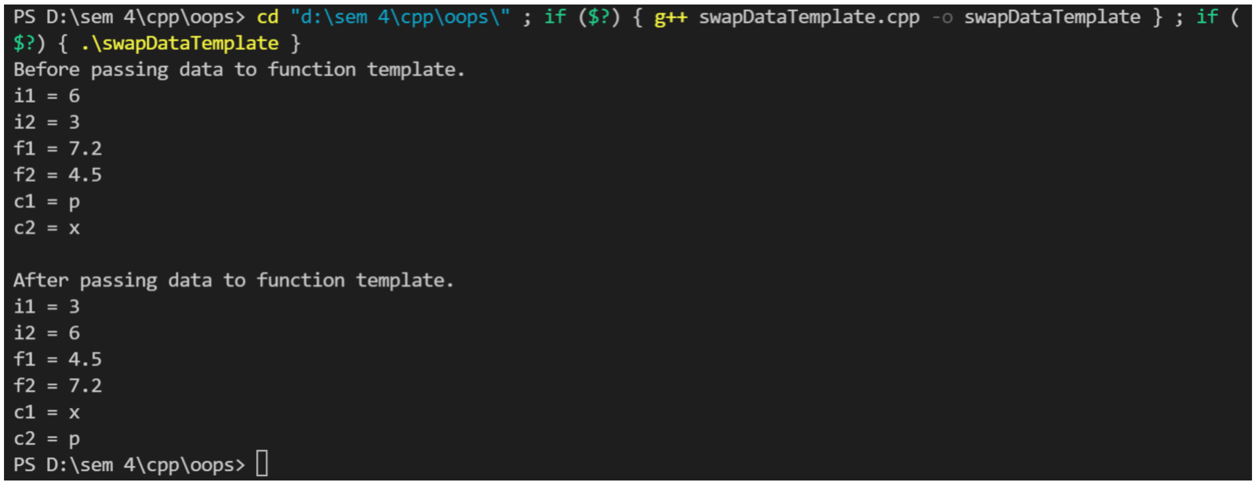
Aim:

Write a program to define the function template for swapping two items of various datatypes such as integers, float and characters.

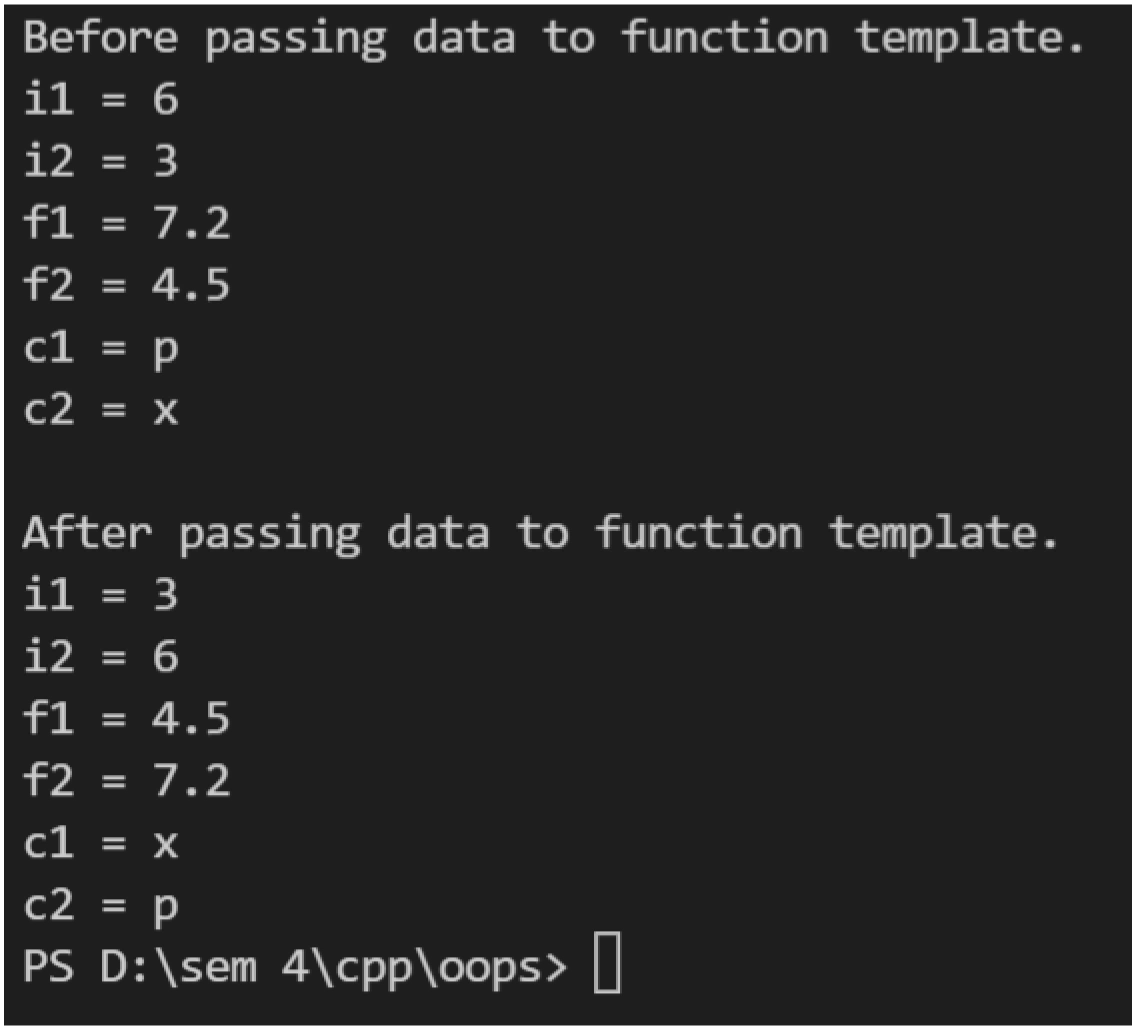
Source Code:

```
#include <iostream>
using namespace std;

template <typename T>
void Swap(T &n1, T &n2)
{
    T temp;
    temp = n1;
    n1 = n2;
    n2 = temp;
}

int main()
{
    int i1 = 6, i2 = 3;
    float f1 = 7.2, f2 = 4.5;
    char c1 = 'p', c2 = 'x';

    cout << "Before passing data to function template.\n";
    cout << "i1 = " << i1 << "\ni2 = " << i2;
    cout << "\nf1 = " << f1 << "\nf2 = " << f2;
    cout << "\nc1 = " << c1 << "\nc2 = " << c2;

    Swap(i1, i2);
    Swap(f1, f2);
    Swap(c1, c2);

    cout << "\n\nAfter passing data to function template.\n";
    cout << "i1 = " << i1 << "\ni2 = " << i2;
    cout << "\nf1 = " << f1 << "\nf2 = " << f2;
    cout << "\nc1 = " << c1 << "\nc2 = " << c2;

    return 0;
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ swapDataTemplate.cpp -o swapDataTemplate } ; if (
$?) { .\swapDataTemplate }
Before passing data to function template.
i1 = 6
i2 = 3
f1 = 7.2
f2 = 4.5
c1 = p
c2 = x

After passing data to function template.
i1 = 3
i2 = 6
f1 = 4.5
f2 = 7.2
c1 = x
c2 = p
PS D:\sem 4\cpp\oops> 
```



```
Before passing data to function template.
i1 = 6
i2 = 3
f1 = 7.2
f2 = 4.5
c1 = p
c2 = x

After passing data to function template.
i1 = 3
i2 = 6
f1 = 4.5
f2 = 7.2
c1 = x
c2 = p
PS D:\sem 4\cpp\oops> 
```

Viva Questions

Q1). What are templates in C++?

Ans.

Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type.

A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept.

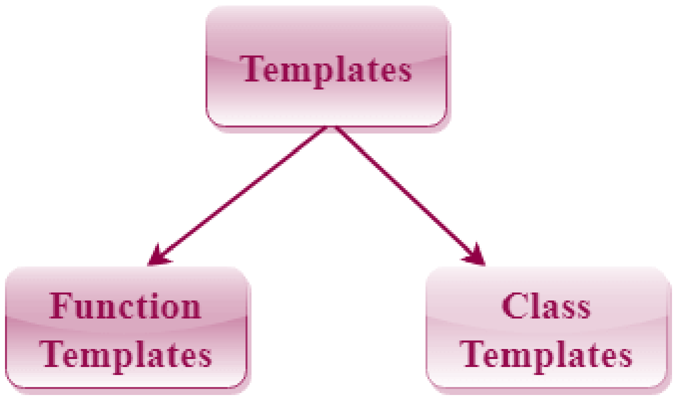
There is a single definition of each container, such as **vector**, but we can define many different kinds of vectors for example, **vector <int>** or **vector <string>**.

Q2). How can templates be classified?

Ans.

Templates can be represented in two ways:

- Function templates
- Class templates



Function Templates:

We can define a template for a function. For example, if we have an add() function, we can create versions of the add function for adding the int, float or double type values.

Class Template:

We can define a template for a class. For example, a class template can be created for the array class that can accept the array of various types such as int array, float array or double array.

Q3). Write about Function templates.

Ans.

- C++ supports a powerful feature known as a template to implement the concept of generic programming.
- A template allows us to create a family of classes or family of functions to handle different data types.
- Template classes and functions eliminate the code duplication of different data types and thus makes the development easier and faster.
- Multiple parameters can be used in both class and function template.
- Template functions can also be overloaded.
- We can also use nontype arguments such as built-in or derived data types as template arguments.

Q4). What are different Data types?

Ans.

Data Type	Meaning	Size (in Bytes)
`int`	Integer	2 or 4
`float`	Floating-point	4
`double`	Double Floating-point	8
`char`	Character	1
`wchar_t`	Wide Character	2
`bool`	Boolean	1
`void`	Empty	0